Mock CCC '23 1 S5 - The Obligatory Data Structures Problem

Time limit: 4.0s **Memory limit:** 1G

Mock CCC wouldn't be a real mock CCC without a data structures problem, would it?

You're given two sequences of N integers a and b, and you need to support two operations:

- `Update(l, r, x)` - set a_l through a_r to all be equal to x.
- `Query(l, r)` - compute the number of indices i where $l \leq i \leq r$ and $a_i \geq b_i$.

DMOJ has been having some issues with test data taking up too much space, so you're going to generate the operations and solve the problem for us!

Constraints

$1 \leq N \leq 10^5$

$1 \leq M \leq 3 \cdot 10^6$

$1 \leq A, B \leq 2^{16}$

$1 \leq a_i, b_i \leq 10^9$

In tests worth 1 mark, $M \leq 50$.

In tests worth an additional 2 marks, $M \leq 10^6$.

In tests worth an additional 4 marks, $M \leq 2 \cdot 10^6$.

Input Specification

The first line contains four integers, N, M, A, and B.

The second line contains N integers, the sequence a.

The third line contains N integers, the sequence b.

Because the number of operations is large, you'll be generating the m operations using the following code.

```
int a = A, b = B, C = ~(1<<31), M = (1<<16)-1;
int rnd(int last) {
  a = (36969 + (last >> 3)) * (a & M) + (a >> 16);
  b = (18000 + (last >> 3)) * (b & M) + (b >> 16);
  return (C & ((a << 16) + b)) % 1000000000;
}
```

The intended solution does not exploit the random number generation used; it would work even if the operations were hand-constructed.

For the ith operation, call `rnd` three times, setting l to be `rnd(last) % n + 1`, then r to be `rnd(last) % n + 1`, then x to be `rnd(last) + 1`. If $l > r$, then swap them. if $l + r + x$ is even, the ith operation is `Query(l, r)`. Otherwise, it is `Update(l, r, x)`.

`last` is always the last return value of `Query`. `last` starts out being `0`.

Output Specification

Let v_i be 0 if the ith operation was an `Update`, and the result of the `Query` operation otherwise. Output $\sum_{i=1}^{m} i \cdot v_i$ modulo `998244353`.

Sample Input

```
5 10 5 6
5 4 5 2 1
1 2 2 4 5
```

Sample Output

```
87
```